Java Exception Handling Concepts(Part 1)

For

BCA 3rd Semester (Session 2017-20) Students (J.D. Women's College, Patna)

By

Sumit Kumar Verma

Exception Handling

Exceptions are those runtime errors which can be handled programmatically in the application. Exception handling adds the features of **robustness** into the language.

Exception handling is used to maintain the Normal Flow of the Program i.e Exception Handling protect from abnormal termination of the program at runtime.

In Java, each runtime error is represented by an object. There exists class JRE for describing runtime errors. At the root of this class hierarchy is an abstract class named **Throwable** which has 2 non-abstract subclasses named **Exception** and **Error**.

Commonly used subclasses of Exception:-

1. ArithmeticException – represents an invalid arithmetical operation such as divide by zero.
   Example-4/0
2. ArrayIndexOutOfBoundsException – represents an attempt to refer non-existent array elements.
   Example-int a[]=new int[5];
   a[5]=5 ;
3. NullPointerException – represents an attempt to use a null containing reference variable for referencing object features.
   Example-String s1=NULL;
   System.out.println(s1.length);
4. NumberFormatException – represents an attempt to correct non-numeric string into a number.
   Example-int a=”india”;
5. etc.

Commonly used subclasses of error –

1. Virtual Machine Error – represents malfunctioning of JRE.
2. StackOverflowError – represents overflowing of stack.
3. NoSuchMethodError – represents an attempt to invoke a non-existent method.
4. NoClassDefFoundError – represents an attempt to load a non-existent class.
5. etc.

Exception handling is facilitated with the help of following keywords:-

a. **try** keyword is used to mark a block of statements as error prone statements i.e we have to write within try block ,those statement which may give error.

```
try
{
        error prone statements
    }
        catch(Exception e)
        {
                Statements to be executed in case of Runtime Error
        }
        }


        OR

try
{
        error prone
    Statements
        }
        finally
        {
    Statements to be executed whether error occurs or not
        }


        OR

try
{
        error prone
    Statements
        }
    catch(Exception e)
    {
        Statements to be executed in case of Error
    }

        finally
        {
    Statements to be executed whether exception occurs or not
        }
```

b. **catch** keyword is used to define an exception handler i.e. it defines an alternative course of action that is taken by the JRE when the error represented by the catch block occurs.

Catch block become silent when Exception does not occurred.

Write a program to demonstrate need of Exception handling.

```
 class Ex1
{
      public static void main(String args[])
      {
         int a=Integer.parseInt(args[0]);
         int b=Integer.parseInt(args[1]);

          System.out.println(a/b);

          System.out.println("normal flow");
        }
}
```

o/p:1>if we pass 4 2 as input then it give

2

normal flow

2>if we pass 4 0 as input then abnormal termination will occurred

```
class Ex1
{
      public static void main(String args[])
      {

         int a=Integer.parseInt(args[0]);
         int b=Integer.parseInt(args[1]);
           try
```

```
            {
            System.out.println(a/b);
             }catch(ArithmeticException e)
              {
                  System.out.println(e);
               }
           System.out.println("normal flow");
        }
}
```

o/p:1>if we pass 4 2 as input then it give

2

normal flow

2>if we pass 4 0 as input then it give

ArithmeticException

normal flow